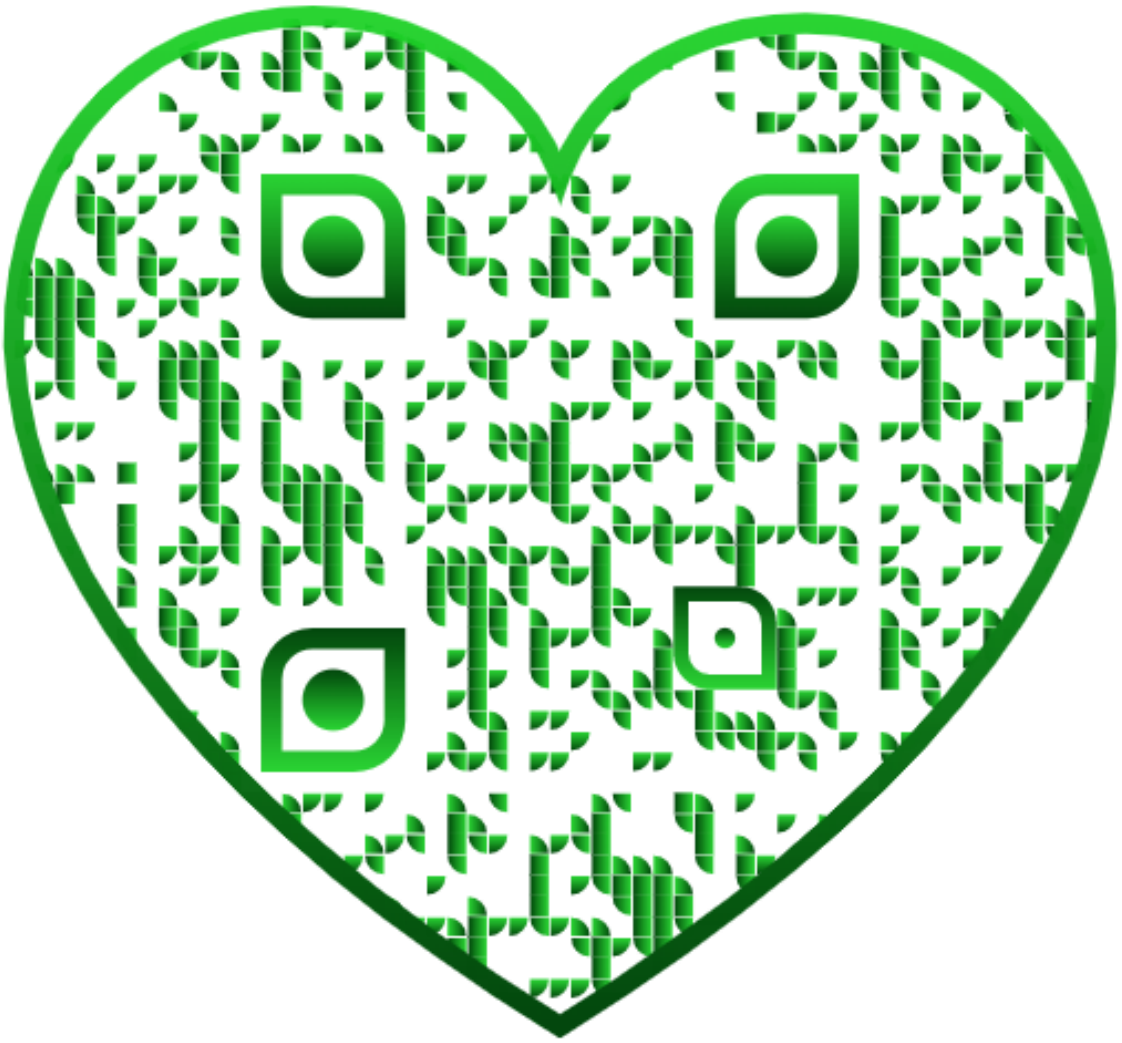
PHP 7 & MySQL

mémento

Cyril Pierre de Geyer
Guillaume Ponçon

avec la contribution
de Stéphane Mariel

EYROLLES



Constantes et variables

La constante « CONSTANT » contient « 10 »

const `CONSTANT = 10;`
echo `CONSTANT;`
Les constantes sont généralement utilisées pour définir les paramètres d'une application web (racine du site, nom de la base de données...). Depuis PHP 5.6, les constantes peuvent contenir des tableaux.

La variable « $variable » contient « 10 »

```
$variable = 10;
echo $variable;
```

Toutes les variables commencent par un dollar (**$**) en PHP.

Opérateurs d’affectation

Affecter une valeur à une variable

```
$string = 'chaîne de caractères !';
$int = 10;
```

Conversion de type (transtypage)

PHP définit dynamiquement le type de la variable. Il est cependant possible de le forcer en utilisant le transtypage :

```
// (int), (float), (string), (bool), (array), (object), (unset)
$int = (int) $var;
```

CONSEIL D'EXPERT Pourquoi convertir le type d'une variable ?

Selon son type, la valeur d'une variable n'aura pas le même comportement.

Lier des chaînes de caractères (concaténation)

```
$nom = 'Openska';
$texte = 'Les formations PHP avec '.$nom;
$texte .= ' partout en France';
// $texte contient "Les formations PHP avec Openska partout en France"
```

Somme et incrémentation

```
$i = 5;
$i += 5; // Correspond à $i = $i + 5; $i contient 10
$i++; // Ajoute 1 à $i, utile dans les boucles
```

Opérateurs de comparaison

Opérateur	Signification	Renvoie TRUE si	Exemple
==	Égal à	**$a** est égal à **$b**	**$a == $b**
===	Identique, en type et valeur	$a est égal à $b et leur type est le même. ("5" est différent de 5)	**$a === $b**
!= ou <>	Différent de	**$a** est différent de **$b**	**$a != $b**
!==	Différent en valeur ou en type	**$a** a une valeur différente de **$b** ou est de type différent	**$a !== $b**
<	Inférieur à	**$a** est plus petit que **$b**	**$a < $b**
<=	Inférieur ou égal à	**$a** est inférieur ou égal à **$b**	**$a <= $b**
>	Supérieur à	**$a** est plus grand que **$b**	**$a > $b**
>=	Supérieur ou égal à	$a est supérieur ou égal à $b	$a >= $b
<=>	Retourne -1, 0 ou 1 suivant que égal ou supérieur à **$b** (PHP 7)	**$a** est inférieur,	**$a <=> $b**

CONSEIL D'EXPERT **Attention aux comparaisons de valeurs et de types !**
Une comparaison avec "==" vérifie que deux valeurs sont égales alors que la même comparaison avec "===" vérifie qu'elles sont identiques.

Conditions

Condition simple : if

```
if (condition1) {
    action
} elseif (condition2) {
    action2
} else {
    action3
}
```

EXEMPLE

```
// mt_rand(0,5) renvoie une valeur aléatoire comprise entre 0 et 5
if (mt_rand(0,5) == 5) {
    echo "La valeur aléatoire est 5";
} else {
    echo "La valeur aléatoire n'est pas 5";
}
```

Condition multiple : switch

```
switch (variable) {
     case <valeur1> :
        // On passe ici si variable == <valeur1>
        break;
     case <valeur2> :
        // Ou ici si variable == <valeur2>
        break;
     default :
     // Sinon on passe là (action par défaut)
}
```

Opérateur ternaire

```
// if ... else ... en une seule instruction
echo isset($var) ? '$var contient ' . $var : '$var n\'existe pas';
```

Opérateur ?? (PHP 7)

```
// Retourne la valeur de $color si non nulle, sinon 'orange'
echo $color ?? 'orange';
```

Boucles

Boucle while

```
while (condition) {
    // action exécutée tant que condition est vraie
}
```

EXEMPLE **Pour boucler de 0 à 9**

```
$i = 0;
while ($i < 10) {
    echo $i++;
}
```

Boucle for

```
for (initialisation; condition; incrementation) {
    // action exécutée tant que condition est vraie
}
```

EXEMPLE Pour boucler de 0 à 9

```
for ($i = 0; $i < 10; $i++) {
    echo $i;
}
```

Boucle foreach : itérer sur un tableau ou une collection

Pour manipuler jusqu'à la n-ième valeur du tableau :

```
foreach ($array_or_collection as $element) {
   // action
}
```

Pour manipuler jusqu'à la n-ième clé *et* valeur du tableau :

```
foreach ($array as $key => $element) {
   // action
}
```

Les superglobales

Les superglobales sont des tableaux prédéfinis accessibles partout dans le code. Elles contiennent des informations utiles sur votre environnement.

Superglobale	Contenu
$_GET	Contient les données envoyées via l'URL (ex : http://www.toto.com/index.php?**id=15**&**nom=titi**).
$_POST	Contient les données envoyées via la méthode POST (généralement utilisée pour les formulaires).
$_FILES	Contient la liste et les caractéristiques des fichiers transmis au serveur par le visiteur.
$_SERVER	Contient des informations sur la requête en cours, sur PHP et sur le serveur HTTP.
$_ENV	Contient des informations « bas niveau » sur votre environnement (système d'exploitation).

PRATIQUE **Accès aux superglobales (cf. formulaires)**

Pour plus de sécurité, préférez l'utilisation de la fonction `filter_input()` plutôt qu'un accès direct aux superglobales `$_ENV`, `$_SERVER`, `$_POST` et `$_GET`.

Déclaration et appel de fonctions

Déclarer une fonction

```
my_function($arg1, $arg2, $arg3) {
// liste d'instructions
}
```

Argument optionnel, avec valeur par défaut

```
function my_function($arg = 12) { ... }
```

Retourner une valeur avec « return »

```
function my_function($arg = 12) {
   if ($arg == 12) {
        return 'La valeur est 12';
     }
     return 'La valeur est différente de 12';
}
```

REMARQUE **Attention à l'ordre des arguments**

Dans le prototype de la fonction, il faut d'abord déclarer les arguments obligatoires (sans valeur par défaut), puis les optionnels, puis le tableau d'arguments (fonction variadique).

Déclaration compatible PHP7

```
// typage fort des arguments et de la valeur de retour
function my_function(int $arg1, array $arg2 = [], ...$args):
string {
   return $arg1 . "\n" . print_r($arg2, true) . print_r($args,
true);
}
// Appel correct de la fonction
echo my_function(1, [2, 'a'], 3, 'quatre');
```

Définir le type des paramètres

Il est possible de déclarer le type des paramètres d'une fonction ou d'une méthode.

```
// function nomFonction(type $var, type $var2)
function add(int $a, int $b){
    var_dump($a, $b);
}
```

Si cette fonction est appelée en lui passant des données de types incorrects, une TypeError sera automatiquement levée par PHP.

Fonctions d’affichage

Afficher simplement un texte

```
echo 'Bonnes pratiques'; // Affiche « Bonnes pratiques »
```

Affichage formaté

```
// Affiche « PHP 7, PHP 7.10, PHP 111 »
printf('PHP %2d, PHP %.2f, PHP %b', 7.1, 7.1, 7.1);
```

Déboguer les variables

```
echo '<pre>';     // amélioration de l'affichage html
print_r($array_or_object);     // affichage condensé
var_dump($array_or_object);     // affichage détaillé
echo '</pre>';
```

Déclarer et afficher une chaîne

```
$cd = 'ef';     // la variable $cd contient 'ef'
echo 'ab$cd';     // Affiche « ab$cd »
echo "ab$cd";     // Affiche « abef » car $cd est interprété
```

PRATIQUE **Guillemets ou apostrophes ?**

Avec des guillemets (`""`), PHP interprète le contenu (et donc remplace les variables par leur valeur et les codes de caractères spéciaux tels que `\n` par leur équivalent). Avec des apostrophes (`''`), PHP ne fait que copier-coller le contenu.

Avec des guillemets, si une variable ne doit pas être interprétée, placer un \ devant :

```
$a = "le contenu de \$age est $age.\n";
```

Typage et arrondis numériques

Vérifier le type d’une valeur

```
is_int($var);   // vrai si $var contient un entier
is_numeric($var);   // vrai si $var est un nombre
is_array($var);    // vrai si $var contient un tableau
is_string($var);   // vrai si $var est une chaîne
is_float($var);    // vrai si nombre à virgule flottante
```

Arrondir, tronquer des valeurs numériques

```
echo ceil(4.2);   // affiche l'entier supérieur (5)
echo floor(4.2);   // affiche l'entier inférieur (4)
echo round(4.2);   // affiche l'entier le plus proche (4)
```

Chaînes de caractères

Taille d’une chaîne

```
echo strlen($str); // Affiche le nombre d'octets de $str
```

Formatage d’une chaîne avec printf et ses dérivés

```
$str = "%'.-25s %d\n";
printf($str, 'Note des lecteurs', 20);
// Affiche : Note des lecteurs........ 20
```

Position d’une chaîne dans une autre

```
echo strpos("abcdef", "cd"); // Position de "cd" dans "abcdef"
```

Remplacement

```
// Remplace 'old' par 'new' dans $str
$str = str_replace('old', 'new', $str);
// Renvoie les quatre premiers caractères à partir du deuxième
$str = substr($str, 2, 4);
```

Nettoyage

Suppression des caractères superflus aux extrémités

```
$clean = trim(" Bonjour ! \n ");
```

Formatage HTML

Conversion des caractères spéciaux en HTML

```
echo htmlspecialchars($str);
```

Redirection d’une page vers une URL

```
header('Location: http://www.openska.com');
```

CONSEIL D’EXPERT **Chaînes de caractères unicode**

Il est recommandé d'utiliser, lorsque c'est possible, un jeu de caractères unicode tel que UTF-8. Si tel est le cas, utilisez impérativement les fonctions multi-octets (**mb_substr**, **mb_strlen**, ...).

```
mb_internal_encoding('UTF-8');
echo mb_strlen("càd"); // affiche 3. strlen() afficherait 4
```

Manipulations HTTP

Redirection d'une page vers une URL

Affichage d'une page HTML unicode

```
header('Content-Type: text/html; charset=UTF-8');
```

Désactivation du cache côté client

```
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT");
```

Dates

Afficher une date

Le timestamp Unix est un entier qui représente une date (nombre de secondes depuis le 1er janvier 1970).

```
/* En ajoutant 3600 * 24 secondes, on ajoute une journée*/
$timestamp_demain = time() + 3600 * 24;
echo date('d/m/Y', $timestamp_tomorrow); // Affiche 02/01/2017
```

Paramètres de formatage de la fonction date

Code	Description	Code	Description
d	Jour du mois (ex : 01)	**j**	Jour du mois (ex : 1)
D	Jour de la semaine (ex : Mon)	**z**	Jour de l'année (ex : 34)
w	Jour de la semaine (numérique)	**W**	Numéro de semaine (ex : 42)
l	Jour de la semaine textuel	**Y**	Année (ex : 2015)
F	Mois textuel (ex : December)	**y**	Année (ex : 15)
m	Mois numérique (ex : 06)	**n**	Mois numérique (ex : 6)
M	Mois en trois lettres (ex : Jan)	**t**	Nombre de jours dans le mois
g	Heure au format 12h (ex : 6)	**G**	Heure au format 24h (ex : 18)
h	Heure au format 12h (ex : 06)	**H**	Heure au format 24h (ex : 06, 18)
i	Minutes (ex : 05)	**s**	Secondes (ex : 02)
c	Date ISO8601	**r**	Date RFC 2822

Construire un timestamp Unix

```
$timestamp = mktime(0,0,0,7,14,2008);   // 14/07/2008
echo date('l', $timestamp) ;    // Affiche « Friday »
```

Vérifier une date

checkdate($month, $day, $year) ; // Renvoie TRUE si la date existe

CONSEIL D'EXPERT **Manipuler des dates en PHP**

Pour manipuler les dates, il existe un objet DateTime très efficace.

Gestion des courriels

Lors de l'envoi de courriels, l'ensemble du traitement se fait par l'intermédiaire d'une seule fonction : **mail()**.

Envoi d'un courriel simple

```
$dest = "cyril@php.net,guillaume@openstates.com";
$subject = "Mon premier courriel";
$body = "Monsieur \nLigne2 \nJe remarque ...";
mail($dest, $subject, $body);
```

PRATIQUE Signaler les sauts de ligne

Pour signaler les sauts de ligne dans le message, il suffit d'utiliser le caractère spécial `\n` et de faire en sorte que ces caractères soient interprétés en délimitant la chaîne avec des guillemets (`"`) et non avec des apostrophes (`'`).

Envoi d’un courriel à plusieurs destinataires en changeant l’expéditeur

```
$dest = 'cyril@openska.com, luc@openska.com';
$subject = 'Vos cotisations !';
$header = "From: responsable@urssaf.fr\n";
$header .= "Reply-to: adresseretour@toto.fr\n";
mail($dest, $subject, "Bonjour ...", $header);
```

Envoi d’un courriel au format HTML

```
$dest = 'cyril@php.net';
$subject = 'Courriel au format HTML';
$body = '<html><body><b>Bonjour</b>...';
$header = "Content-Type: text/html";
mail($dest, $subject, $body, $header);
```

Gestion des formulaires

Formulaire minimal en HTML

```
<form method="post" action="form.php">
  <input type="text" name="email">
  <input type="submit" name="ok" value="OK">
</form>
```

Récupération des informations en PHP

```
// Tableau associatif contenant les variables passées par l'URL (GET)
$_GET[]
// Tableau associatif contenant les variables passées par POST
$_POST[]
// Tableau contenant les fichiers envoyés via le formulaire
$_FILES[]
```

PRATIQUE **Filtrer et valider**

Utilisez `filter_input_array()` pour récupérer les valeurs. Exemple :

```
$values = filter_input_array(INPUT_POST);
```

Balise HTML pour l'upload de fichier

```
<form action="recept.php" method="POST"
          enctype="multipart/form-data">
<input type="file" name="fichier">
```

Récupération du fichier en PHP

```
$file = $_FILES['fichier']['name'];
$size = $_FILES['fichier']['size'];
$tmp = $_FILES['fichier']['tmp_name'];
$type = $_FILES['fichier']['type'];
$error = $_FILES['fichier']['error'];
```

move_uploaded_file($tmp, 'nouveau_nom');

Attention à filtrer et valider toutes les données entrantes.

Balises HTML pour passer un tableau en paramètre

```
<input type="text" name="person[firstname]">
<input type="text" name="person[lastname]">
```

Affichage du tableau passé en paramètre

```
$person = filter_input(INPUT_GET, 'person', FILTER_DEFAULT,
FILTER_REQUIRE_ARRAY);
echo 'Nom : ', $person;
```

Les sessions

Les sessions permettent de conserver les informations concernant un utilisateur tout au long de sa navigation (ex. : panier d’achat, espace sécurisé…). Les sessions sont gérées côté serveur contrairement aux cookies qui le sont côté client. Notez qu'il existe des outils comme Redis ou Memcache pour stocker les sessions.

Initialiser une session

```
session_start();
```

ATTENTION Ne rien envoyer au navigateur avant l'initialisation des sessions.

Mettre une variable dans la session

```
$_SESSION['auteurs'] = "Guillaume et Cyril";
var_dump($_SESSION);
echo 'hello '.$_SESSION['auteurs'].' !<br />';
```

Fermer une session

```
session_destroy();
```

Gestion des tableaux

Déclarer un tableau

```
$tab = array();
$tab2 = array(1, 2, "chaine", 45.3);
$tab3 = [1, 2, 'chaine']; // PHP7
```

Ajouter des valeurs à un tableau

```
$tab[] = "Guillaume";    // clé implicite : 0
$tab[] = "Cyril";   // clé implicite : 1
```

Tableau associatif (clés et valeurs liées)

```
// $tab['clé'] = "valeur";
$tab['nom'] = "Ponçon";
$tab['prenom'] = "Guillaume";
```

Connaître le nombre d’éléments d’un tableau

```
$nb = count($tab);   // nombre d’éléments
```

Rechercher un élément dans un tableau

Recherche d'une clé à partir d'une valeur

```
$cle = array_search('Guillaume', $tab);
```

Vérifier si un tableau contient une valeur

```
// Est-ce qu'il y a « Linux » dans le tableau $tab ?
if (in_array('Linux', $tab)) {
    echo 'Trouvé';
}
```

Trier un tableau

```
sort($tab);   // trie un tableau
asort($tab);   // trie un tableau en conservant la liaison clé-valeur
rsort($tab);   // trie un tableau en ordre inverse
```

Supprimer une valeur

```
unset($tab['prenom']); // suppression
```

Une variable est-elle un tableau ?

```
if (is_array($tab)) {... // est-ce un tableau ?
```

Boucler sur un tableau

```
foreach ($tab as $key => $value) {
    echo "$key : $value\n";
}
```

Gestion des fichiers

Récupération express du contenu d’un fichier ou d’une URL

```
$cFile = file_get_contents('fichier.txt');
```

Écriture express dans un fichier

```
$var = 'PHP est une plate-forme de développement';
file_put_contents('fichier.txt', $var, APPEND_FILE);
```

CONSEIL La constante `APPEND_FILE` permet d'ajouter du contenu en fin de fichier. Par défaut, `file_put_contents()` remplace le contenu du fichier.

Lecture bufferisée d'un fichier

```
// Évite de charger tout le fichier en mémoire
$fp = fopen('monfichier.txt', 'r');
while (!feof($fp)) {
    echo fgets($fp, 1024);
}
fclose($fp);
```

Fonctions de manipulation de fichiers

Fonction	Effet
fopen()	Permet d'ouvrir un fichier. La fonction renvoie un descripteur à utiliser pour les traitements suivants.
fgets()	Renvoie une ligne du fichier. Prend en paramètre le file descriptor du fichier et le nombre maximum de caractères à lire.
fread()	Renvoie *n* caractères d'un fichier.
fwrite()	Écrit dans un fichier.
fclose()	Ferme la connexion au fichier.

Modes d’ouverture de fichiers

Mode	Signification
r	Ouvre en lecture seule ; place le pointeur en début de fichier.
r+	Ouvre en lecture et écriture ; place le pointeur en début de fichier.
w	Ouvre en écriture seule ; place le pointeur au début du fichier et réduit la taille du fichier à 0. Si le fichier n’existe pas, il est créé.
w+	Ouvre en lecture et écriture ; place le pointeur au début du fichier et réduit la taille du fichier à 0. Si le fichier n’existe pas, il est créé.
a	Ouvre en écriture seule ; place le pointeur à la fin du fichier. Si le fichier n’existe pas, il est créé.
a+	Ouvre en lecture et écriture ; place le pointeur à la fin du fichier. Si le fichier n’existe pas, il est créé.

Fonctions d’information

Fonction	Effet
feof($descr) et **feof($descr)**	Test de fin de ligne/fin de fichier
file_exists($nom_fichier)	Existence d’un fichier
filesize($nom_fichier)	Taille d’un fichier
stat($nom_fichier)	Informations diverses sur un fichier

Exploiter un fichier de configuration (INI)

- PHP offre un moyen simple d'utiliser des fichiers de configuration classique de syntaxe similaire au php.ini.
- Lecture d'un fichier .ini

```
$tab = parse_ini_file('monfichier.ini');
```

CONSEIL D'EXPERT

Vous pouvez utiliser SplFileInfo et SplFileObject si vous êtes à l'aise avec la POO.

Bases de données

L’extension PDO permet de se connecter à de nombreux SGBD du marché.

Connexion à la base de données MySQL

La chaîne **$dsn** *(Data Source Name)*, contient le type de base de données, son nom et l'adresse du serveur sur laquelle elle est hébergée.

```
$dsn = "mysql:host=sql.openstates.com;dbname=users";
$login = "user"; $pass = "Il0v3OpenSk@";
// Connexion à la base de données
$pdo = new PDO($dsn, $login, $pass);
// Lancer une exception en cas d'erreur
$pdo->setAttribute(PDO::ATTR_ERRMODE,
          PDO::ERRMODE_EXCEPTION);
```

Exécuter une requête SELECT

```
$sql = "SELECT name FROM users LIMIT 0, 10";
$users = $pdo->query($sql);
while ($user = $users->fetch()){
    echo $user['name'];
}
```

Exécuter une insertion paramétrée

```
$sql = "INSERT INTO user (name, email) VALUES (?, ?)";
$stmt = $pdo->prepare($sql);
$new_user = ['Guillaume','contact@openstates.com'];
$stmt->execute($newUser);
```

PERFORMANCES ET SÉCURITÉ

Les requêtes paramétrées (ou préparées) permettent d'exécuter plusieurs fois la même requête avec des données différentes et d'optimiser ainsi les performances et la sécurité (contre les injections SQL par exemple).

Gestion des transactions

```
$pdo->beginTransaction();
try {          // Exécuter les requêtes ici
    $pdo->commit();    // Validation de toutes les requêtes
} catch (PDOException $e) {
    $pdo->rollback();    // Annulation de toutes les requêtes
}
```

Programmation orientée objet

Composition d'une classe

```
class ConfigManager extends Config {
     private    $host;
     protected    $port;
     public    $name;
   public function __construct(...) { ... }
}
```

- **ConfigManager** est le nom de la classe.
- **Config** est le nom de la classe mère, précédé du mot-clé **extends**. Les propriétés et méthodes **public** et **protected** de la classe mère sont accessibles dans la classe fille **ConfigManager**.
- **private**, **protected** et **public** sont appelées « portées ».
- **$host**, **$port** et **$name** sont les propriétés de la classe **ConfigManager**.
- **__construct** est une méthode de la classe **ConfigManager**.

Portée (ou visibilité) des attributs (propriétés et méthodes)

- **private** : accessible uniquement dans la classe courante.
- **protected** : accessible dans la classe courante et ses classes filles.
- **public** : accessible partout.

TERMINOLOGIE OBJET

- Propriétés : variables d'une classe.
- Méthodes : fonctions d'une classe.
- Constructeur **__construct** : méthode spéciale appelée lors de l'instanciation de la classe (appel avec **new**).
- Destructeur **__destruct** : méthode spéciale appelée lors de la désinstanciation de la classe (suppression avec **unset** ou en fin de script).

Manipuler une classe

- Instanciation d'une classe

```
$configManager = new ConfigManager();
```

- Appel d'une méthode

```
$value = $configManager->getValue('host');
```

Espaces de noms (namespaces)

Les espaces de noms permettent d'éviter le risque de conflit de nommage : quand on définit une classe, on la positionne dans son espace de noms.

```
<?php
namespace OdtPhp;
// Espace de noms pour toutes les classes de
// la bibliothèque OdtPHP : templating document LibreOffice
class Document
{
// Classe représentant un document
}
```

Chargement automatique (autoload)

PHP propose un système de chargement automatique des classes (lazy loading). Grâce à ce mécanisme, seuls les fichiers contenant les classes utilisées sont inclus.

```
function recherche_declaration($nomClasse)
{
// On implémente la recherche
}
spl_autoload_register('recherche_declaration');
```

Ajouter dans chargement automatique (autoload) l'exemple suivant :

Déclaration du chargement automatique

```
// Chargement des fichiers contenant des classes
spl_autoload_register(function($className) {
    include strtr($className, '\\', '/') . '.php';
});

// L'inclusion de OpenStates/Config.php est automatique $config =
new OpenStates\Config;
```

Les expressions rationnelles

Également appelées expressions régulières, elles sont utilisées pour effectuer toutes sortes de manipulations sur les chaînes de caractères.

CONSEIL Utiliser les expressions rationnelles avec précaution
Utiliser en priorité les fonctions natives de manipulation de chaînes (**substr**, **strtr**, etc.). Les expressions rationnelles sont pratiques mais pas toujours performantes.

Symboles utilisés dans les expressions rationnelles

Symbole	Description
^	Marque le début d'une chaîne
$	Marque la fin d'une chaîne
|	Marque l'alternative (ou)
-	Indique un intervalle. Ex : [a-d] = entre a et d (a, b, c ou d)
()	Capture de motif utilisé pour « matcher » un contenu.
[]	Classes de caractères. Ex : [0-9a-f] = un chiffre hexadécimal
[[:x:]]	Classe prédéfinie (x = alpha, digit, space, alnum, lower, upper, xdigit, blank, print)
.	Tout caractère
?	Facultatif (0 ou 1 occurrence)
*****	Facultatif (0 ou plusieurs occurrences)
+	Obligatoire (1 ou plusieurs occurrences)
{x}	Doit apparaître exactement x fois (x étant un nombre)
{x,}	Doit apparaître au moins x fois
{x,y}	Doit apparaître entre x et y fois

Fonctions PCRE usuelles

Fonction	Description
preg_match($regex, $str)	Recherche le motif dans la chaîne.
preg_match_all($regex, $str)	Recherche toutes les occurrences du motif dans la chaîne.
preg_replace($regex, $replacement, $str)	Effectue un remplacement sur toutes les occurrences correspondant au motif dans la chaîne.

Vérifier la validité syntaxique d’une adresse e-mail

```
$mail = 'guillaume.poncon@openstates.com'
$exp = '/^[a-z0-9\._-]+@[a-z0-9\._-]+\.[a-z]{2,4}$/i';
echo preg_match($exp, $mail);
```

Extraire les balises HTML

```
$html = '<b>Best practices <i>PHP</i></b><br />';
$exp = '|<([^>]+) */?>|U';
preg_match_all($exp, $html, $out);
```

Administration SQL

Les exemples qui suivent sont compatibles avec MySQL. Avec d'autres SGBD, ces syntaxes peuvent être différentes.

Créer une base de données

```
CREATE DATABASE base;
```

Créer une table

```
CREATE TABLE client(idClient int(11), nom char(60));
```

On peut utiliser le paramètre **AUTO_INCREMENT** sur un champ pour avoir une clé unique numérique.

```
CREATE TABLE client (
   idClient int(11) NOT NULL auto_increment,
   nom char(60),
   PRIMARY KEY (idClient)
) ;
```

Modifier la structure d'une table

```
ALTER TABLE table
     ADD INDEX [nom_index] (nom_champ,...)
ou
     ADD PRIMARY KEY (nom_champ,...)
ou
     ADD UNIQUE [nom_index] (nom_champ,...)
ou
     ADD FULLTEXT [nom_index] (nom_champ,...)
```

Supprimer une table

```
DROP TABLE [IF EXISTS] table
```

Insertion de données

Insertion standard

INSERT [INTO] nom_de_table [(nom_colonne,...)]
VALUES (),...
Si un champ est défini comme étant auto-incrémenté, il suffit de lui donner la valeur **NULL** pour que MySQL se débrouille tout seul.
INSERT client VALUES (NULL, 'BOURDON')

Insertion via une sous-requête

```
INSERT [INTO] nom_de_table [(nom_colonne,...)]
SELECT...
```

Insertion non complète via SET

```
INSERT [INTO] nom_de_table SET
nom_colonne=(expression)...
```

Insertion multiple en une passe

```
INSERT INTO table (champ1, champ2) VALUES
('val11', 'val12'), ('val21', 'val22'), ('val31', 'val32')
```

Modification et suppression de données

Modifier des données (UPDATE)

```
UPDATE nom_de_table
SET nom_colonne1=expr1 [, nom_colonne2=expr2, ...]
[WHERE where_definition]
```

Pour mettre à jour un enregistrement, indiquer sa clé primaire.

```
UPDATE client SET nom='PIERRE' WHERE idClient=3;
```

Remplacer des données (REPLACE)

```
REPLACE
     [INTO] nom_de_table [(nom_de_colonne,...)]
     VALUES (expression, ...), (...), ...
ou
REPLACE
     [INTO] nom_de_table [(nom_de_colonne,...)]
     SELECT ...
ou
REPLACE
     [INTO] nom_de_table
     SET nom_de_colonne=expression,
         nom_de_colonne=expression,...
```

NOTE Il s'agit d'une commande spécifique à MySQL. Avec d'autres SGBD, **REPLACE** fonctionne différemment.

Supprimer des données (DELETE)

```
DELETE FROM nom_de_table
     [WHERE clause_where]
     [ORDER BY ...]
     [LIMIT lignes]
```

Récupération de données

Récupérer des données (SELECT)

```
SELECT champ
     [FROM table_1]
     [WHERE condition]
```

Sélectionner l’ensemble des champs d’une table

```
SELECT * FROM table
```

Nommer un champ avec « AS »

```
SELECT MIN(nom_champ) AS minimum FROM table
```

Trier avec « ORDER BY »

```
SELECT * FROM table ORDER BY champ [DESC|ASC]
```

Limiter le nombre de lignes résultat avec « LIMIT »

```
SELECT * FROM table LIMIT nombre
          # LIMIT <à partir de>, <nombre de lignes>
SELECT * FROM table LIMIT 5, 10
```

Compter le nombre de lignes

```
SELECT count(*) AS nbr FROM table WHERE condition
```

Retenir la valeur minimale ou maximale

```
SELECT MIN(nom_champ) AS minimum from table
SELECT MAX(nom_champ) AS maximum from table
```

Enlever les doublons (DISTINCT)

```
SELECT DISTINCT champ FROM table WHERE condition
```

Filtrer avec la clause WHERE

Syntaxe générale de WHERE

```
[SELECT | UPDATE | DELETE]
WHERE condition
ou
WHERE champ IN | NOT IN ('valeur1', 'valeur2', 'valeur3')
ou
WHERE champ BETWEEN 'limite1' AND 'limite2'
ou
WHERE champ LIKE 'expression'
```

Les outils pour développer en PHP

Éditeurs spécialisés

- NetBeans IDE
- Eclipse + PDT
- PHPStorm ou ZendStudio

Accélérateur

- OPcache (en standard depuis PHP 5.5)

Débogueurs et profileurs

- Advanced PHP Debugger (APD)
- Xdebug + Kcachegrind/Wincachegrind
- PHP Debugger DBG
- Xhprof

ERREURS CLASSIQUES ET CONSEILS

- Parse error : vérifiez s'il ne manque pas un ; aux lignes précédant la ligne indiquée dans le message d'erreur.
- Cannot send session cache limiter - headers already sent : attention, ne faites pas d'echo et ne laissez pas les retours à la ligne avant l'appel à session_start(), header(), etc.
- Call to a member function on a non-object : vous essayez de faire appel à une méthode dont l'objet n'existe pas. Vérifiez que vous avez bien instancié votre objet.
- Failed opening required "lib.php" (include_path=".:/var/www/lib") : PHP n'a pas réussi à ouvrir un fichier (généralement une bibliothèque). Vérifiez que votre fichier est bien présent. Vous pouvez aussi utiliser la directive de configuration include_path (dans votre fichier **php.ini**).

MÉTHODE DE RÉSOLUTION D'UN PROBLÈME

Quelques pistes

- Affichez la requête SQL et vérifiez-la dans un outil tel que PHPMyAdmin.
- Utilisez var_dump() pour afficher le contenu et le type de votre variable.
- Vérifiez que vous êtes bien connecté à votre base de données.

Méthode générique

- Faites un copier-coller du message d'erreur dans votre moteur de recherche.
- Rendez-vous sur www.php.net et faites une recherche sur la fonction qui pose problème. Consultez les commentaires des utilisateurs.
- Allez sur un forum (ex : www.phpfrance.com) et faites une recherche. Si aucune réponse ne convient, postez un message en indiquant bien le problème et son contexte.
- Rendez-vous sur le channel IRC #phpfrance sur un serveur undernet.

INFO Travailler avec les versions 7.x de PHP

De nombreuses modifications ont été apportées à PHP 7, aussi bien en termes de performances que de fonctionnalités. Les frameworks n'étant parfois plus compatibles avec les versions de PHP 5 antérieures, il est préférable d'utiliser dès maintenant PHP 7.

© Groupe Eyrolles, 2017

ISBN : 978-2-212-67402-6

Attention : pour lire les exemples de lignes de code, réduisez la police de votre support au maximum.

Pour suivre toutes les nouveautés numériques du Groupe Eyrolles, retrouvez-nous sur Twitter et Facebook

@ebookEyrolles

EbooksEyrolles

Et retrouvez toutes les nouveautés papier sur

@Eyrolles

Eyrolles

This eBook was posted by AlenMiler on AvaxHome!
Many New eBooks in my Blog: http://avxhome.in/blogs/AlenMiler
Mirror: https://avxhome.unblocked.tw/blogs/AlenMiler